DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on April 3, 2008 has been entered.

Response to Amendment

2. **Claims 1-2**, **5-8**, **12**, and **16-19**, have been amended.

Claims 11, and **22-24**, have been canceled.

Claims 25-28 have been added.

Rejections for the newly added **claims 1-10**, **12-21**, and **25-28**, are presented in this instant Office action.

Response to Arguments

3. Applicant's arguments with respect to **claims 1-10**, **12-21**, and **25-28**, have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an

> international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) and the Intellectual Property and High Technology Technical Amendments Act of 2002 do not apply when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. Therefore, the prior art date of the reference is determined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

5. **Claims 1-3**, **5-8**, **12-14**, **16-19**, and **25-28**, are rejected under 35 U.S.C. 102(e) as being anticipated by Rawat et al. (*Pat. No. 6,662,340, filed on May 30, 2002; hereinafter Rawat*).

Regarding **claim 1**, Rawat clearly shows and discloses a computer-implemented method for dynamic data type enrichment (*Abstract*) comprising the steps:

loading an application program in to memory (*Client-side program code examines electronic documents such as web pages and automatically fills in fields of forms contained in the document with the appropriate data from a user profile, without requiring prior mapping or examination of the form*, [Column 3, Lines 61-65]. *It is well inherent that any computer program is loaded into computer's memory in advance to execution*), the application program comprising a variable that is defined as an instance of both a basic

data type and a specific data type (*Figure 1 shows the 'Credit Card Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number*);

accessing metadata at runtime to map the variable to a definition of the specific data type (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301*, [Column 5, Lines 18-22]); and

processing the variable consistently with the metadata definition of the specific data type (*in which the fields of an HTML form are identified and mapped to the correct user data based on visible form elements such as field labels. Following mapping and identification, the fields of the form are populated with the correct user data, without reference to a previous, stored mapping or analysis of the form, and without requiring user intervention*, [Colum 4, Lines 56-62]).

Regarding **claim 2**, Rawat further discloses the application program uses an application programming interface for accessing the metadata (*locating the field labels, associating them with a field, and then mapping*

the field to the correct metadata based on a best match from a field label dictionary 304, [Column 5, Lines 18-22]).

Regarding **claim 3**, Rawat further discloses the application program calls through the application programming interface at least one metadata service that relates to the basic data type (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data*, [Column 5, Lines 44-50]).

Regarding **claim 5**, Rawat further discloses the basic data type is defined in one or more of Visual Basic, Java, or C++ (*other commonly known scripting and programming languages would also be suitable for the invention such as VBSCRIPT, PERL, JAVA, or JPYTHON*, [Column 5, Lines 58-61]).

Regarding **claim 6**, Rawat further discloses the metadata defines an allowed value range for the specific data type that is a subset of an allowed value range for the basic data type (*Figure 1 shows the specific data type of credit card number requires 16 digits, each digit is from the range 0 to 9 defined by the basic data type, i.e. integers that makes up the credit card sequence*).

Regarding **claim 7**, Rawat further discloses the metadata defines a text label for a user input field (*if a field lacks a label, the algorithm may*

analyze the field's programmatic name. Following field name analysis, the field name is compared to the entries in the Field label dictionary and a match found. The field is then mapped as described above, [Column 7, Lines 19-23]).

Regarding **claim 8**, Rawat further discloses the variable is set to a value entered into the user input field (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data*, [Column 5, Lines 44-50]).

Regarding **claim 12**, Rawat clearly shows and discloses a computer system ([Column 5, Lines 3-5]) comprising:

a memory (*laptop or desktop comprises memory*, [Column 5, Lines 3-5]) storing an application program that comprises a variable that is defined as an instance of both a basic data type and a specific data type (*Figure 1 shows the 'Credit Card Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number*); and

a processor executing instructions (*laptop or desktops comprises CPU to execute instructions loaded on memory*, [Column 5, Lines 3-5]) to:

access metadata at runtime to map the variable to a definition of the specific data type (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301*, [Column 5, Lines 18-22]); and

process the variable consistently with the metadata definition of the specific data type (*in which the fields of an HTML form are identified and mapped to the correct user data based on visible form elements such as field labels. Following mapping and identification, the fields of the form are populated with the correct user data, without reference to a previous, stored mapping or analysis of the form, and without requiring user intervention*, [Colum 4, Lines 56-62]).

Regarding **claim 13**, Rawat further discloses an application programming interface to access the metadata from the application program (*locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304*, [Column 5, Lines 18-22]).

Regarding **claim 14**, Rawat further discloses the application programming interface provides at least one metadata service that relates to the basic data type used by the application program (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data*, [Column 5, Lines 44-50]).

Regarding **claim 16**, Rawat further discloses the basic data type is defined in one or more of Visual Basic, Java, or C++ (*other commonly known scripting and programming languages would also be suitable for the invention such as VBSCRIPT, PERL, JAVA, or JPYTHON*, [Column 5, Lines 58-61]).

Regarding **claim 17**, Rawat further discloses the metadata defines an allowed value range for the specific data type that is a subset of an allowed value range for the basic data type (*Figure 1 shows the specific data type of credit card number requires 16 digits, each digit is from the range 0 to 9 defined by the basic data type, i.e. integers that makes up the credit card sequence*).

Regarding **claim 18**, Rawat further discloses the metadata defines a text label for a user input field (*if a field lacks a label, the algorithm may analyze the field's programmatic name. Following field name analysis, the*

field name is compared to the entries in the Field label dictionary and a match found. The field is then mapped as described above, [Column 7, Lines 19-23]).

Regarding **claim 19**, Kesler further discloses the variable is set to a value entered into the user input field (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data*, [Column 5, Lines 44-50]).

Regarding **claim 25**, Rawat clearly shows and discloses the variable is declared as an instance of both the basic data type and the specific data type at design time (*Figure 1 shows the 'Credit Card Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number*).

Regarding **claim 26**, Rawat clearly shows and discloses a computer-implemented method for dynamic data type enrichment (*Abstract*) comprising:

> loading an application program into memory (*Client-side program code examines electronic documents such as web pages and automatically fills in fields of forms contained in the document with the appropriate data from a user profile, without requiring prior mapping or examination of the form*, [Column 3, Lines 61-65]. *It is*

well inherent that any computer program is loaded into computer's memory in advance to execution), the application program comprising a variable that is defined as an instance of both a basic data type and a specific data type (*Figure 1 shows the 'Credit Card Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number*);

accessing metadata over a network at runtime using an application programming interface (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301*, [Column 5, Lines 18-22]), the metadata mapping the variable to a definition of the specific data type that indicates a text label for an input field to the variable (*if a field lacks a label, the algorithm may analyze the field's programmatic name. Following field name analysis, the field name is compared to the entries in the Field label dictionary and a match found. The field is then mapped as described above*, [Column 7, Lines 19-23]) and indicating allowed value range for the variable (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the*

client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data, [Column 5, Lines 44-50])); and

processing the variable consistently with the indicated allowed value range (*in which the fields of an HTML form are identified and mapped to the correct user data based on visible form elements such as field labels. Following mapping and identification, the fields of the form are populated with the correct user data, without reference to a previous, stored mapping or analysis of the form, and without requiring user intervention*, [Colum 4, Lines 56-62]).

Regarding **claim 27**, Rawat clearly shows and discloses a computer program product comprising instructions embodied on a memory of a computer system that cause at least one processor of the computer system to execute a method ([Column 11, Lines 59-64]), the method comprising:

loading an application program into memory (*Client-side program code examines electronic documents such as web pages and automatically fills in fields of forms contained in the document with the appropriate data from a user profile, without requiring prior mapping or examination of the form*, [Column 3, Lines 61-65]. *It is well inherent that any computer program is loaded into computer's*

memory in advance to execution), the application program comprising a variable that is defined as an instance of both a basic data type and a specific data type (*Figure 1 shows the 'Credit Card Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number*);

accessing metadata at runtime to map the variable to a definition of the specific data type (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301*, [Column 5, Lines 18-22]); and

processing the variable consistently with the metadata definition of the specific data type (*in which the fields of an HTML form are identified and mapped to the correct user data based on visible form elements such as field labels. Following mapping and identification, the fields of the form are populated with the correct user data, without reference to a previous, stored mapping or analysis of the form, and without requiring user intervention*, [Colum 4, Lines 56-62]).

Regarding **claim 28**, Rawat further discloses the metadata is stored on a hard disk, and the application program loaded into memory accesses the metadata by retrieving the metadata from the hard disk (*Figures 3 & 4*).

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. **Claims 4**, and **15**, are rejected under 35 U.S.C. 103(a) as being unpatentable over Rawat et al. (*Pat. No. 6,662,340, filed on May 30, 2002; hereinafter Rawat*) in view of Koseki et al. (*Pat. No. US 6,732,124, filed on February 9, 2000; hereinafter Koseki*).

Regarding **claim 4**, and **15** Rawat does not explicitly disclose a metadata service copies the metadata to a metadata cache.

Koseki discloses a metadata cache is provided as part of the computer's main memory to hold a copy of metadata objects from the metadata volume. A metadata loading unit reads out a specific metadata object from the metadata volume to the metadata cache when a transaction demands it ([Column 9, Lines 48-58]).

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Koseki with the teachings of Rawat for the purpose of preventing unwanted changes in the original metadata by modifying the copy of metadata in the metadata cache instead of directly manipulating the original in the metadata volume ([Column 9, Lines 55-58] of Koseki).

8. **Claims 9-10**, and **20-21**, are rejected under 35 U.S.C. 103(a) as being unpatentable over Rawat et al. (*Pat. No. 6,662,340, filed on May 30, 2002; hereinafter Rawat*) in view of Logan et al. (*Pub. No. US 2003/0093790, filed on June 8, 2002; hereinafter Logan*).

Regarding **claims 9-10**, and **20-21**, Rawat does not explicitly disclose the metadata is stored along with the application program in private instance of the metadata. Also, Rawat does not explicitly disclose storing metadata in public instance of the metadata.

Logan discloses that metadata contributed by other users and stored in a public database as well as private database. The metadata stored may be created, edited and deleted using a Web server or other server operates to contribute to the metadata ([0308]). Logan further discloses the playback control 211 of the Personal Video Recorder (PVR) controls the playback of stored video programming seen at 217, stored electronic program guide (EPG) data seen at 218, application data such as standard templates stored at 220, metadata describing programs and

program segments stored at 221, and other system control data stored at 222 ([0301]-[0304])

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Logan with the teachings of Rawat for the purpose of enhancing user's enjoyment of available broadcast programming content by utilizing metadata created either at a central location for shared used by connected users, or at each individual user's location ([Abstract] of Logan).

Conclusion

9. These following prior arts made of record and not relied upon are considered
pertinent to applicant's disclosure:

Chamberlain et al. (Pat. No. US 6,427,236) teaches method for installing a patch based on patch criticality and software execution format.

Forbes et al. (Pat. No. US 6,381,742) teaches software package management.

McGuire et al. (Pat No. US 6,493,871) teaches method and system for downloading updates for software installation.

The Examiner requests, in response to this Office action, support(s) must be shown for language added to any original claims on amendment and any new claims. That is, indicate support for newly added claim language by specifically

pointing to page(s) and line no(s) in the specification and/or drawing figure(s). This will assist the Examiner in prosecuting the application.

When responding to this office action, Applicant is advised to clearly point out the patentable novelty which he or she thinks the claims present, in view of the state of the art disclosed by the references cited or the objections made. He or she must also show how the amendments avoid such references or objections See 37 CFR 1.111(c).

Contact Information

10. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Son T. Hoang whose telephone number is (571) 270-1752. The Examiner can normally be reached on Monday - Friday (7:30 AM – 5:00 PM).

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Christian Chace can be reached on (571) 272-4190. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-

direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/S.H./
Son T. Hoang
Patent Examiner
Art Unit 2165
April 17, 2008

/S. P./
Primary Examiner, Art Unit 2164

/Christian P. Chace/
Supervisory Patent Examiner, Art Unit 2165